**„Programozás”  
komplex beadandó feladat**

Készítette: *Heki*Neptun-azonosító: \*\*\*\*\*\*E-mail: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Kurzuskód: IT-18PROGEGGyakorlatvezető neve: Menyhárt László Gábor Dr.

2025. január 9.

Tartalom

[Felhasználói dokumentáció 3](#_Toc187285617)

[Feladat 3](#_Toc187285618)

[Futási környezet 3](#_Toc187285619)

[Használat 3](#_Toc187285620)

[A program indítása 3](#_Toc187285621)

[A program használata billentyűzetről való bevitel esetén 3](#_Toc187285622)

[A program használata fájlból való bevitel esetén 3](#_Toc187285623)

[A program kimenete 4](#_Toc187285624)

[Minta bemenet és kimenet 4](#_Toc187285625)

[Hibalehetőségek 4](#_Toc187285626)

[Fejlesztői dokumentáció 6](#_Toc187285627)

[Feladat 6](#_Toc187285628)

[Tervezés 6](#_Toc187285629)

[Specifikáció 6](#_Toc187285630)

[Visszavezetés 6](#_Toc187285631)

[Algoritmus 6](#_Toc187285632)

[Fejlesztői környezet 7](#_Toc187285633)

[Forráskód 7](#_Toc187285634)

[Megoldás 7](#_Toc187285635)

[Függvénystruktúra 7](#_Toc187285636)

[A kód 8](#_Toc187285637)

[A kód (magas szintű függvényekkel) 11](#_Toc187285638)

[Tesztelés 16](#_Toc187285639)

[Automatikus tesztek (Bíró) 17](#_Toc187285640)

[Automatikus tesztek (Bíró, magas szintű függvényekkel) 17](#_Toc187285641)

[Érvényes tesztesetek 18](#_Toc187285642)

[Érvénytelen tesztesetek 19](#_Toc187285643)

[Fejlesztési lehetőségek 20](#_Toc187285644)

2. Felhasználói dokumentáció
   1. Feladat

**A települések legalább felében melegedő napok**

A meteorológiai intézet az ország N településére adott M napos időjárás előrejelzést, az adott településen az adott napra várt legmagasabb hőmérsékletet.  
Készíts programot, amely megadja azokat a napokat, amikor legalább a települések felében melegedés várható az előző naphoz képest!

* 1. Futási környezet

IBM PC, exe futtatására alkalmas, 64-bites operációs rendszer (pl. Windows 11). Nem igényel egeret.

* 1. Használat
     1. A program indítása

A program a \*\*\*\*\*\*\\*verzio\*\bin\Debug\net6.0\KomplexMagas.exe néven található a tömörített állományban.

* + 1. A program használata billentyűzetről való bevitel esetén

Az \*\*\*\*\*\*.exe fájl elindításával a program az adatokat a **billentyűzet**ről olvassa be a következő sorrendben:

|  |  |  |
| --- | --- | --- |
| # | Adat | Magyarázat |
| **1.** | *Települések száma (n) …* | 1 és 1000 közötti egész szám |
| **1.** | *Napok száma (m)* | 1 és 1000 közötti egész szám |
| **2.** | *1. település m nap adatai felsorova* | -50 és 50 közötti egész szám |
| **3.** | *2. település m nap adatai felsorova* | -50 és 50 közötti egész szám |
| **…** | *…* | … |
| **n+1** | *n. település m nap adatai felsorova* | -50 és 50 közötti egész szám |

* + 1. A program használata fájlból való bevitel esetén

Lehetőségünk van az adatokat **fájl**ban is megadni. Ekkor a programot *parancssorban* a következőképpen kell indítani, feltételezve, hogy a bemeneti fájlok mellette helyezkednek el:

\*verzionev\*.exe < be1.txt

A fájl felépítésének a következő formai követelményei vannak. A fájl első sorában a települések száma (n) és a napok száma (m) van. A következő n sor mindegyikében m darab szám szerepel, közülük az i-edik sorban a j-edik szám az i-edik településben a j-edik nap maximum hőmérséklete. Például:

4 3

1 0 4

2 0 1

0 2 0

1 0 0

* + 1. A program kimenete

A program kiírja azoknak a napoknak a darabszámát és a sorszámait, ahol az adott napon több mint a települések felében melegedés volt az előző naphoz képest.  
Például:

1 3

* + 1. Minta bemenet és kimenet

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

* + 1. Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha bármelyik megadandó adat nem egész szám vagy a megadott intervallumon belül van. Hiba esetén a program azzal jelzi a hibát, hogy újra bekéri az adott adatrészt.

* + - 1. Mintafutás hibás bemeneti adatok esetén:

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

1. Fejlesztői dokumentáció
   1. Feladat

**A települések legalább felében melegedő napok**

A meteorológiai intézet az ország N településére adott M napos időjárás előrejelzést, az adott településen az adott napra várt legmagasabb hőmérsékletet.  
Készíts programot, amely megadja azokat a napokat, amikor legalább a települések felében melegedés várható az előző naphoz képest!

* 1. Tervezés
     1. [Specifikáció](https://progalap.elte.hu/specifikacio/?data=H4sIAAAAAAAACpWSzU7CQBCAX2XSU0kW7RaqshETTZAY%2Fg6oB0sPFRZTShdCizESErnBe%2BjB5%2BBNeBJnlxZakYNNZnfnZ2e%2Bme1MC8e86%2FW9rht5I6Ex7YYzEJvlskkg2G7PPOCCR1Kx6cmJILgETmfSEW2XgS%2FtUql5eNEdj%2Fwk0Fcxt68MIj7k4%2BmQhw08vITvns%2BgCW8o%2BSuoExkG%2BB2G6R6BQQ7KCYMtdcgDdeAyY9uVQgJ1m%2Fd4yLYFmrsCaaeu8jYq1fbTetFo1XWvrJo7SpGTWSp9BhQ2q08QaqWGYcD6O4yNQda4WX54OA01NYfpqA5iNXAY6HnLUPGZRpTFMrblHrCc7pPtXCVv7e5xvai3qtf3%2BqBsYh5y0NRm9QW6ODVzanQySyIa0SIeRqHG7JnWcyMXn1swKEhfwMCSe8zCwE6GZlODALVQTBR5NhwCey9FS0rMrFfeOdsLeqVPLfgTmHJXzWFBDLWcmFO4AUc6PE14OB1GGqNz8hd08Rg01iJQIFB0ds9vYxPIcE7gImUsxV1JfjNDV%2FhFp7L9B44egSua6RkVS2mtYGUYjDTDseLO%2FAeU0Tu6yAMAAA%3D%3D)

**Be**: n∈N, m∈N, bemenet∈N[1..n,1..m]

**Sa**: k∈N

**Ki**: napok∈N[1..k]

**Fv**: telepulesMelegszik: N x N -> L,

telepulesMelegszik(i, j) = bemenet[i, j - 1] < bemenet[i, j]

**Fv**: napMelegedes: N -> N,

napMelegedes(j) = **DARAB**(i=1..n, telepulesMelegszik(i, j))

**Ef**: 1 ≤ n ≤ 1000 és 1 ≤ m ≤ 1000 és ∀i∈[1..n]:(∀j∈[1..m]: (-50 ≤ bemenet[i, j] ≤ 50))

**Uf**: (k, napok) = **KIVÁLOGAT**(j=2..m, napMelegedes(j) ≥ (n/2), j)

* + 1. Visszavezetés

*Megszámolás*

e..u ~ 1..n

T(i) ~ telepulesMelegszik(i, j)

*Kiválogatás*

e..u ~ 2..m

T(i) ~ DARAB(i=1..n, telepulesMelegszik(i, j)) ≥ (n/2)

f(i) ~ j

* + 1. [Algoritmus](https://progalap.elte.hu/stuki/?data=H4sIAAAAAAAACq1Wa2%2FaMBT9K5UrVZvKWAgJr45J0MegHeqD0tEiPji2A4GQ0BBerfjvs1N8cQZojdp8iTm%2BPufcB0lekUNRCWnpYsHUtYxhGhkzb2YKGZRCfX%2FGgjrf9qaum0IT5jISMrpBPJ%2ByCSq9ova197z0C3diHfEBkEJjHDAv3BzqO5QyD5Vs7E5YCoXLMeMHJux5yjzC%2BAHSd1zKz9Qp5%2B6gXvN63Lr5dc53ZiNy1wgqc760%2BlPvtNC6RN1VahMi5ZUzG3nV1B4TIQ7ZiIfziJAtQg55eOwPS%2BVOF60UfamjGPqgDtfQhMTydnjdtAgFCQBiEpB%2B8npeZa0%2FF4%2F3hO%2BMbttZq14TZXKN9tNVplWL6gns0oQilyTPOVeOHPgedULH5wFoUNbT6ZHooE%2BXzbXLeixT7qAe5nLnpmGBAwBiDpQEkpYBF3rLxU27ESUMPFJOIVbklF68P2GnnEmnvV0JQ07cAXRFOlDalMTBjtGi1nq2Xh7vXxZXVgM0AIhpQGWSF7Vat8Y3jn4WFbVqTO%2Byg6oDcgDE5C5DZsxz4XSvnOf7Y2EfAiWfcvKT7EuDwj7wSDmFWJFTBnO3nGP%2FMxCW6A8LO05q8C3TPXLDEwXpir4FU1YNsEf6sXFRuhUp7AiBiojyy15A%2BSUQ86%2BwJpio42MhUSG%2FZ5Vg%2FgQSAMQk4NmSvCOTerNxPnx44TsV1%2BxXKo%2BnUXOGjTOy0OzNkxiAmPLEadXsgfn039mCQMmnnPycTMCgsA88Uk4hTvJv35otah0c9cKTgy%2Fed%2F3rDyv4uX%2BclF7tGycogqiQ7ARUaNMaxbLCmuwd2xl2y4NoqGSfYagk8GGhoRhbkQsvCQn9XoBHojev3MFIHGlgRzyoZzhwsOWKTxvEfwa%2Bv%2FW%2Bm%2FT9%2BcMmbK0s0BrDlAVraNUV7XH9IPpKenPRQYe2rfGL0xyKuy2ayDECmKWZsLKjVV7LajhaEYbtQh5135KqYjLsBf7Uo2tiS7d0ESbukpgRibE8wXJlRytqEI3pgk58zG3REX5FqvySdLYuMWZTso9OqfEWq7jWOb%2BL37aLxbddTSuQoilWjDGD5tSKrFZ%2FAYzmyR3NCgAA)

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

* 1. Fejlesztői környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows 11 Home). Visual Studio 2022 (Version 17.2.3) fejlesztői környezet.

* 1. Forráskód

A teljes fejlesztői anyag –kicsomagolás után– a \*\*\*\*\*\* nevű könyvtáron belül a KomplexMagas és KomplexAlap mappákban található meg. A fej­lesztés során használt könyvtár-struktúra:

|  |  |
| --- | --- |
| Állomány | Magyarázat |
| \*\*\*\*\*\*\KomplexAlap\bin\Debug\net6.0\KomplexAlap.exe  \*\*\*\*\*\*\KomplexMagas\bin\Debug\net6.0\KomplexMagas.exe | futtatható kód (a futtatáshoz szükséges fájlokkal) |
| \*\*\*\*\*\*\KomplexAlap\obj\  \*\*\*\*\*\*\KomplexMagas\obj\ | mappa fordításhoz szükséges kódokkal |
|  |  |
| \*\*\*\*\*\*\KomplexAlap\Program.cs | C# Alap megoldás forráskód |
| \*\*\*\*\*\*\KomplexMagas\Program.cs | C# Magas megoldás forráskód |
|  |  |
| \*\*\*\*\*\*\be1.txt | teszt-bemeneti fájl1 |
| \*\*\*\*\*\*\be2.txt | teszt-bemeneti fájl2 |
| \*\*\*\*\*\*\be3.txt | teszt-bemeneti fájl3 |
| \*\*\*\*\*\*\be4.txt | teszt-bemeneti fájl4 |
| \*\*\*\*\*\*\be5.txt | teszt-bemeneti fájl5 |
|  |  |
| \*\*\*\*\*\*\doksi\\*\*\*\*\*\*.docx | dokumentációk (ez a fájl) |

* 1. Megoldás
     1. Függvénystruktúra
     2. A kód

A Program.cs fájl tartalma:

|  |
| --- |
| /\*  Készítette: Heki  Neptun: \*\*\*\*\*\*  E-mail: \*\*\*\*\*\*  Feladat: A települések legalább felében melegedő napok  \*/  using System;  using System;  namespace komplex  {  internal class Program  {  const int MaxN = 1000;  static void Beolvasas(ref int telepulesSzam, ref int napokSzama, out int[,] adatok)  {  adatok = new int[MaxN, MaxN];  if (!Console.IsInputRedirected)  {  //első sor  bool telepulesbool, napokbool, hosszbool = false;  Console.Error.WriteLine("Kerem a [telepulesek szamat] [napok Szamat]\tV:");  string[] tmp = Console.ReadLine().Split();  while (tmp.Length != 2)  {  Console.Error.WriteLine("Nem 2 adatot adtál meg, [telepulesek szamat] [napok Szamat]\tV:");  tmp = Console.ReadLine().Split();  }  telepulesbool = Int32.TryParse(tmp[0], out telepulesSzam);  napokbool = Int32.TryParse(tmp[1], out napokSzama);  if (!(1 <= telepulesSzam && telepulesSzam <= 1000 && 1 <= napokSzama && napokSzama <= 1000 && napokbool && telepulesbool && tmp.Length == 2))  {  while (!(1 <= telepulesSzam && telepulesSzam <= 1000 && 1 <= napokSzama && napokSzama <= 1000 && napokbool && telepulesbool && tmp.Length == 2))  {  Console.Error.WriteLine("Hibas adatok kerlek add meg ujra [1 <= N <= 1000] [1 <= M <= 1000]\tV: ");  tmp = Console.ReadLine().Split();  if (tmp.Length == 2)  {  telepulesbool = Int32.TryParse(tmp[0], out telepulesSzam);  napokbool = Int32.TryParse(tmp[1], out napokSzama);  }  }  }  //összes többi sor  string[] tmplista = new string[napokSzama];  adatok = new int[telepulesSzam, napokSzama];  for (int i = 0; i < telepulesSzam; i++)  {  Console.Error.WriteLine($"Kerem a(z) {i + 1}. telepules adatait ({napokSzama} nap)\tV: ");  tmplista = Console.ReadLine().Split();  bool valid = false;  while (!valid)  {  bool tmpvalid = true;  if (tmplista.Length == napokSzama)  {  for (int j = 0; j < napokSzama; j++)  {  bool b = Int32.TryParse(tmplista[j], out adatok[i, j]);  if (!b)  {  tmpvalid = false;  }  else if (b)  {  if (!(-50 <= Int32.Parse(tmplista[j]) && Int32.Parse(tmplista[j]) <= 50))  {  tmpvalid = false;  }  }  }  }  else  {  tmpvalid = false;  }  if (!tmpvalid)  {  Console.Error.WriteLine($"Hibas adat(ok) a(z) {i + 1} telepules adataiban add meg ujra ({napokSzama} nap, -50 <= M[i] <= 50) \tV: ");  tmplista = Console.ReadLine().Split();  }  else if (tmpvalid)  {  valid = true;  }  }  for (int j = 0; j < napokSzama; j++)  {  adatok[i, j] = Int32.Parse(tmplista[j]);  }  }  }  else if (Console.IsInputRedirected)  {  string[] tmp = Console.ReadLine().Split();  telepulesSzam = Int32.Parse(tmp[0]);  napokSzama = Int32.Parse(tmp[1]);  string[] tmplista = new string[napokSzama];  adatok = new int[telepulesSzam, napokSzama];  for (int i = 0; i < telepulesSzam; i++)  {  tmplista = Console.ReadLine().Split();  for (int j = 0; j < napokSzama; j++)  {  adatok[i, j] = Int32.Parse(tmplista[j]);  }  }  }  }  static void feladat(int napokSzama, int telepulesSzam, int[,] adatok, out int[] eredmenyek, ref int eredmenyszam)  {  eredmenyszam = 0;  eredmenyek = new int[napokSzama - 1];  //a 2. naptól kezdem és megszámolom, hogy hányszor melegedett az előző naphoz képest.  for (int i = 1; i < napokSzama; i++)  {  int db = 0;  for (int j = 0; j < telepulesSzam; j++)  {  if (adatok[j, i - 1] < adatok[j, i])  {  db++;  }  }  //megnézem hogy a félénél többször melegedett -e  if (db \* 2 >= telepulesSzam)  {  eredmenyek[eredmenyszam] = i + 1;  eredmenyszam++;  }  }  }  static void kiiras(int[] eredmenyek, int eredmenyszam)  {  Console.Write($"{eredmenyszam}");  for (int i = 0; i < eredmenyszam; i++)  {  Console.Write($" {eredmenyek[i]}");  }  }  static void Main(string[] args)  {  //deklaracio  int telepulesSzam = 0;  int napokSzama = 0;  int[,] adatok;  int eredmenyszam = 0;  int[] eredmenyek;  //beolvasas  //Beolvasas(ref telepulesSzam, ref napokSzama, out adatok);  Beolvasas(ref telepulesSzam, ref napokSzama, out adatok);  //feladat  feladat(napokSzama, telepulesSzam, adatok, out eredmenyek, ref eredmenyszam);  //kiiras  kiiras(eredmenyek, eredmenyszam);  }  }  } |

A kód (magas szintű függvényekkel)

A Program.cs fájl tartalma:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Numerics;  using System.Text;  using System.Threading.Tasks;  namespace komplex  {  internal class Program  {  const int MaxN = 1000;  static void Main(string[] args)  {  // Deklaráció  int telepulesSzam = 0;  int napokSzama = 0;  int[,] adatok;  // Beolvasás  Beolvasas(ref telepulesSzam, ref napokSzama, out adatok);  // Feldolgozás  int[] eredmenyek = Mintak.Kivalogat(  1, napokSzama - 1,  i => TelepulesekTobbsegeNovekedett(i, telepulesSzam, adatok),  i => i + 1  );  // Kiírás  Kiiras(eredmenyek);  }  static void Beolvasas(ref int telepulesSzam, ref int napokSzama, out int[,] adatok)  {  adatok = new int[MaxN, MaxN];  if (!Console.IsInputRedirected)  {  //első sor  bool telepulesbool, napokbool, hosszbool = false;  Console.Error.WriteLine("Kerem a [telepulesek szamat] [napok Szamat]\tV:");  string[] tmp = Console.ReadLine().Split();  while (tmp.Length != 2)  {  Console.Error.WriteLine("Nem 2 adatot adtál meg, [telepulesek szamat] [napok Szamat]\tV:");  tmp = Console.ReadLine().Split();  }  telepulesbool = Int32.TryParse(tmp[0], out telepulesSzam);  napokbool = Int32.TryParse(tmp[1], out napokSzama);  if (!(1 <= telepulesSzam && telepulesSzam <= 1000 && 1 <= napokSzama && napokSzama <= 1000 && napokbool && telepulesbool && tmp.Length == 2))  {  while (!(1 <= telepulesSzam && telepulesSzam <= 1000 && 1 <= napokSzama && napokSzama <= 1000 && napokbool && telepulesbool && tmp.Length == 2))  {  Console.Error.WriteLine("Hibas adatok kerlek add meg ujra [1 <= N <= 1000] [1 <= M <= 1000]\tV: ");  tmp = Console.ReadLine().Split();  if (tmp.Length == 2)  {  telepulesbool = Int32.TryParse(tmp[0], out telepulesSzam);  napokbool = Int32.TryParse(tmp[1], out napokSzama);  }  }  }  //összes többi sor  string[] tmplista = new string[napokSzama];  adatok = new int[telepulesSzam, napokSzama];  for (int i = 0; i < telepulesSzam; i++)  {  Console.Error.WriteLine($"Kerem a(z) {i + 1}. telepules adatait ({napokSzama} nap)\tV: ");  tmplista = Console.ReadLine().Split();  bool valid = false;  while (!valid)  {  bool tmpvalid = true;  if (tmplista.Length == napokSzama)  {  for (int j = 0; j < napokSzama; j++)  {  bool b = Int32.TryParse(tmplista[j], out adatok[i, j]);  if (!b)  {  tmpvalid = false;  }  else if (b)  {  if (!(-50 <= Int32.Parse(tmplista[j]) && Int32.Parse(tmplista[j]) <= 50))  {  tmpvalid = false;  }  }  }  }  else  {  tmpvalid = false;  }  if (!tmpvalid)  {  Console.Error.WriteLine($"Hibas adat(ok) a(z) {i + 1} telepules adataiban add meg ujra ({napokSzama} nap, -50 <= M[i] <= 50) \tV: ");  tmplista = Console.ReadLine().Split();  }  else if (tmpvalid)  {  valid = true;  }  }  for (int j = 0; j < napokSzama; j++)  {  adatok[i, j] = Int32.Parse(tmplista[j]);  }  }  }  else if (Console.IsInputRedirected)  {  string[] tmp = Console.ReadLine().Split();  telepulesSzam = Int32.Parse(tmp[0]);  napokSzama = Int32.Parse(tmp[1]);  string[] tmplista = new string[napokSzama];  adatok = new int[telepulesSzam, napokSzama];  for (int i = 0; i < telepulesSzam; i++)  {  tmplista = Console.ReadLine().Split();  for (int j = 0; j < napokSzama; j++)  {  adatok[i, j] = Int32.Parse(tmplista[j]);  }  }  }  }  static bool TelepulesekTobbsegeNovekedett(int nap, int telepulesSzam, int[,] adatok)  {  int db = 0;  for (int i = 0; i < telepulesSzam; i++)  {  if (adatok[i, nap - 1] < adatok[i, nap])  {  db++;  }  }  return db \* 2 >= telepulesSzam;  }  static void Kiiras(int[] eredmenyek)  {  Console.Write($"{eredmenyek.Length}");  for (int i = 0; i < eredmenyek.Length; i++)  {  Console.Write($" {eredmenyek[i]}");  }  }  }  public static class Mintak  {  #region Összegzés  #region Általános összegzés  public static G Szumma<H, G>(int e, int u, Func<int, H> f, G kezd, Func<G, H, int, G> add)  {  G s = kezd;  for (int i = e; i <= u; i++)  {  s = add(s, f(i), i);  }  return s;  }  public static G Szumma<H, G>(int e, int u, Func<int, H> f, G kezd, Func<G, H, G> add)  {  return Szumma(e, u, f, kezd, (s, p, i) => add(s, p));  }  #endregion  #region Intervallum számok összegzésre  public static int Szumma(int e, int u, Func<int, int> f)  {  return Szumma(e, u, f, 0, (s, p) => s + p);  }  public static double Szumma(int e, int u, Func<int, double> f)  {  return Szumma(e, u, f, 0.0, (s, p) => s + p);  }  #endregion  #region Tömb  public static G Szumma<H, G>(H[] arr, G kezd, Func<G, H, G> add)  {  return Szumma(0, arr.Length - 1, i => arr[i], kezd, (s, p, i) => add(s, p));  }  #endregion  #region Tömb számok összegzésre  public static int Szumma<H>(H[] arr, Func<H, int> f)  {  return Szumma(arr, 0, (s, p) => s + f(p));  }  public static int Szumma(int[] arr)  {  return Szumma(arr, p => p);  }  public static double Szumma<H>(H[] arr, Func<H, double> f)  {  return Szumma(arr, 0.0, (s, p) => s + f(p));  }  public static double Szumma(double[] arr)  {  return Szumma(arr, p => p);  }  #endregion  #endregion  #region Kiválogatás  #region Intervallum  public static H[] Kivalogat<H>(int e, int u, Func<int, bool> t, Func<int, H> f)  {  return Szumma(e, u, f, new List<H>(), (s, p, i) => {  if (t(i))  {  s.Add(p);  }  return s;  }).ToArray();  }  #endregion  #region Tömb  public static G[] Kivalogat<H, G>(H[] arr, Func<H, bool> t, Func<H, int, G> f)  {  return Kivalogat(0, arr.Length - 1, i => t(arr[i]), i => f(arr[i], i));  }  public static H[] Kivalogat<H>(H[] arr, Func<H, bool> t)  {  return Kivalogat(arr, p => t(p), (p, i) => p);  }  #endregion  #endregion  }  } |

* 1. Tesztelés
     1. Automatikus tesztek (Bíró)[[1]](#footnote-1)

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

* + 1. Automatikus tesztek (Bíró, magas szintű függvényekkel)

A képen szöveg, képernyőkép, menü, Betűtípus látható

Automatikusan generált leírás

* + 1. Érvényes tesztesetek
       1. teszteset: be1.txt

|  |
| --- |
| Bemenet – *nincs település, 0 nap* |
| 0 0 |
| Kimenet |
| 0 |

* + - 1. teszteset: be2.txt

|  |
| --- |
| Bemenet – *1 település, 1 nap, nincs ilyen nap* |
| 1 1  1 |
| Kimenet |
| 0 |

* + - 1. teszteset: be3.txt

|  |
| --- |
| Bemenet – *3 tekepülés, 5 nap, 2 nap* |
| 3 5  10 15 12 10 10  11 11 11 11 20  12 16 16 16 20 |
| Kimenet |
| 2 2 5 |

* + - 1. teszteset: be4.txt

|  |
| --- |
| Bemenet – *…* |
| 3 4  1 2 3 4  2 3 2 5  3 4 5 6 |
| Kimenet |
| 3 2 3 4 |

* + - 1. teszteset: be5.txt

|  |
| --- |
| Bemenet – *…* |
| 2 3  5 4 3  6 5 4 |
| Kimenet |
| 0 |

* + 1. Érvénytelen tesztesetek

Billentyűzetes bevitel esetén

* + - 1. teszteset

|  |
| --- |
| Bemenet – *szöveges adat* |
| N = 11tizenegy |
| Kimenet |
| Hibas adat(ok) a(z) {i} telepules adataiban add meg ujra (1 nap, -50 <= M[i] <= 50) V: |

* + - 1. teszteset

|  |
| --- |
| Bemenet – *-50-nél kisebb negatív szám* |
| N = -51 |
| Kimenet |
| Hibas adat(ok) a(z) 1 telepules adataiban add meg ujra (1 nap, -50 <= M[i] <= 50) V: |

* + - 1. 8. teszteset

|  |
| --- |
| Bemenet – *50-nél nagyobb pozitív szám* |
| N = 51 |
| Kimenet |
| Hibas adat(ok) a(z) 1 telepules adataiban add meg ujra (1 nap, -50 <= M[i] <= 50) V: |

* 1. Fejlesztési lehetőségek

1. Többszöri futtatás megszervezése
2. Települések és napok és hónapok nevének megadása
3. Grafikus visszajelzés a számolás lépéseiről

1. Példaképpen egy Mesteres képernyőkép van itt megadva. A végső beadandóban a Bíróból kell a képet kivágni, méghozzá az Eredmény menüpontból kiválasztva a megfelelő „Próbát”. [↑](#footnote-ref-1)